

[Advent of Code](#)
[\[About\]](#)
[\[AoC++\]](#)
[\[Events\]](#)
[\[Settings\]](#)
[\[Log Out\]](#)
[icecreambean 26*](#)
[int y=2016;](#)
[\[Calendar\]](#)
[\[Leaderboard\]](#)
[\[Stats\]](#)
[\[Sponsors\]](#)

--- Day 14: One-Time Pad ---

In order to communicate securely with Santa while you're on this mission, you've been using a **one-time pad** that you **generate** using a pre-agreed algorithm. Unfortunately, you've run out of keys in your one-time pad, and so you need to generate some more.

To generate keys, you first get a stream of random data by taking the **MD5** of a pre-arranged **salt** (your puzzle input) and an increasing integer index (starting with **0**, and represented in decimal); the resulting MD5 hash should be represented as a string of lowercase hexadecimal digits.

However, not all of these MD5 hashes are keys, and you need **64** new keys for your one-time pad. A hash is a key only if:

- It contains three of the same character in a row, like **777**. Only consider the first such triplet in a hash.
- One of the next **1000** hashes in the stream contains that same character five times in a row, like **77777**.

Considering future hashes for five-of-a-kind sequences does not cause those hashes to be skipped; instead, regardless of whether the current hash is a key, always resume testing for keys starting with the very next hash.

For example, if the pre-arranged salt is **abc**:

- The first index which produces a triple is **18**, because the MD5 hash of **abc18** contains **...cc38887a5...**. However, index **18** does not count as a key for your one-time pad, because none of the next thousand hashes (index **19** through index **1018**) contain **88888**.
- The next index which produces a triple is **39**; the hash of **abc39** contains **eee**. It is also the first key: one of the next thousand hashes (the one at index 816) contains **eeeeee**.
- None of the next six triples are keys, but the one after that, at index **92**, is: it contains **999** and index **200** contains **99999**.
- Eventually, index **22728** meets all of the criteria to generate the **64**th key.

So, using our example salt of **abc**, index **22728** produces the **64**th key.

Given the actual salt in your puzzle input, what index produces your **64**th one-time pad key?

Your puzzle answer was **15168**.

--- Part Two ---

Of course, in order to make this process **even more secure**, you've also implemented **key stretching**.

Key stretching forces attackers to spend more time generating hashes. Unfortunately, it forces everyone else to spend more time, too.

Our **sponsors**
help make AoC
possible:

Infi - Fvzcry
gbpu? Xbz
arkg-yriry
glcra va
Hgerpug bc baf
areqxjnegvre!

To implement key stretching, whenever you generate a hash, before you use it, you first find the MD5 hash of that hash, then the MD5 hash of that hash, and so on, a total of `2016` additional hashings. Always use lowercase hexadecimal representations of hashes.

For example, to find the stretched hash for index `0` and salt `abc`:

- Find the MD5 hash of `abc0`: `577571be4de9dcce85a041ba0410f29f`.
- Then, find the MD5 hash of that hash: `eec80a0c92dc8a0777c619d9bb51e910`.
- .
- Then, find the MD5 hash of that hash: `16062ce768787384c81fe17a7a60c7e3`.
- .
- ...repeat many times...
- Then, find the MD5 hash of that hash: `a107ff634856bb300138cac6568c0f24`.
- .

So, the stretched hash for index `0` in this situation is `a107ff...`. In the end, you find the original hash (one use of MD5), then find the hash-of-the-previous-hash `2016` times, for a total of `2017` uses of MD5.

The rest of the process remains the same, but now the keys are entirely different. Again for salt `abc`:

- The first triple (`222`, at index `5`) has no matching `22222` in the next thousand hashes.
- The second triple (`eee`, at index `10`) hash a matching `eeeeee` at index `89`, and so it is the first key.
- Eventually, index `22551` produces the `64`th key (triple `fff` with matching `ffffff` at index `22859`).

Given the actual salt in your puzzle input and using `2016` extra MD5 calls of key stretching, what index now produces your `64`th one-time pad key?

Your puzzle answer was `20864`.

Both parts of this puzzle are complete! They provide two gold stars: **

At this point, you should [return to your advent calendar](#) and try another puzzle.

Your puzzle input was `qzyelonm`.

You can also [\[Share\]](#) this puzzle.